УДК 681.51:57

А.С. КУЛИК, А.Г. ЧУХРАЙ, А.Ю. ЗАВГОРОДНИЙ

Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Украина

НЕЧЕТКИЙ ПОИСК ПОХОЖИХ СТРОК В СИСТЕМАХ ПОВЫШЕНИЯ КАЧЕСТВА ДАННЫХ АВТОМАТИЗИРОВАННЫХ СИСТЕМ ОРГАНИЗАЦИОННОГО УПРАВЛЕНИЯ

Проблема обнаружения дублирующейся информации часто встречается в системах повышения качества данных информационных систем. В работе предлагается новый критерий похожести двух строк, учитывающий возможность использования оператором аббревиатур и сокращений. На базе такого критерия разработаны эффективные методы поиска похожих строк. На основании данных о штатных должностях университета «ХАИ» проведены вычислительные эксперименты, показавшие высокое быстродействие предложенных методов.

качество данных, критерий похожести строк, метод поиска похожих строк

Введение

Эффективность использования информационных систем управления (ИСУ) во многом зависит от качества накапливаемых системой данных. Поэтому создание эффективных средств повышения качества данных становится необходимым условием для успешного функционирования информационных систем. Как показывает практика, наибольшая часть проблем качества данных возникает на этапах ввода данных и интеграции неоднородных данных из различных источников [1, 2]. Одной из ключевых проблем качества, которая возникает как на первом из упомянутых этапов, так и на втором, является дублирующаяся информация. В то же время, в связи с тем, что дублирующиеся данные могут быть искажены друг относительно друга, автоматический поиск таких дубликатов является нетривиальной задачей.

Следует отметить, что задача поиска данных, предположительно представляющих одну и ту же сущность реального мира, достаточно давно привлекает внимание исследователей. Так, в работе [3] представлен метод SoundEx, позволяющий путем фонетического кодирования обнаруживать похожие по звучанию английские слова. В работах [4, 5] исходные строки разбиваются на q-граммы, и в качестве меры похожести двух строк принимается количество q-грамм, присутствующих в обеих строках. Наконец, особого внимания заслуживает подход, базирующийся на расстоянии Левенштейна. В рамках данного подхода похожими считаются строки, для которых количество операций редактирования необходимое для преобразования одной строки в другую не превышает некоторого допустимого порога, что наилучшим образом соответствует ошибкам, допускаемым операторами на этапе ввода данных [2, 6, 7]. Тем не менее, ни один из перечисленных методов не учитывает широко распространенную ситуацию, когда операторы используют аббревиатуры и сокращения, и две похожие строки представляют полное наименование и сокращение. Данная статья посвящена созданию методов, которые бы позволял учитывать такую ситуацию, также как и возможность ошибок операторов при вводе данных.

Критерий похожести двух строк

Для формирования критерия похожести двух строк, который бы позволял обнаруживать различные аббревиатуры и сокращения, необходимо, основываясь на вербальных описаниях, дать формальные математические определения некоторых понятий.

Пусть $\{c_1, c_2, ..., c_a\}$ – конечное непустое множество символов алфавита Σ_c. Будем называть *словом* любую цепочку символов алфавита Σ_c. Пусть $\{delim_1, delim_2, \ldots, delim_b\}$ – конечное непустое множество символов алфавита Σ_{del}, таких что если $x \in \Sigma_{del}$, то $x \notin \Sigma_c$. Тогда любую строку *st*, полученную путем конкатенации символов алфавита $\Sigma=\Sigma_c \cup \Sigma_{del}$, можно представить как *словосочетание* следующим образом:

$$st=w_1 \bullet z_1 \bullet w_2 \bullet z_2 \bullet \ldots \bullet z_{k-1} \bullet w_k, \quad (1)$$

где $w_1, w_2, \ldots, w_k$ – слова, входящие в строку *st*; $z_1, z_2, \ldots, z_{k-1}$ – произвольные цепочки символов алфавита Σ_{del}, $\bullet$ – обозначение операции конкатенации двух цепочек.

Будем считать *сокращением* слова $w=c_1 \bullet c_2 \bullet \ldots \bullet c_m$ такую строку *p*, для которой выполняется следующее условие:

$$F(p,w)=S_2[p \in \{\varepsilon, \varepsilon \bullet c_1 \bullet z, \varepsilon \bullet c_1 \bullet c_2 \bullet z, \ldots, \varepsilon \bullet c_1 \bullet c_2 \bullet \ldots \bullet c_m \bullet z\}] \quad (2)$$

где ε – пустой символ, $z \in \sum_{del}^0 \cup \sum_{del}^1$ т.е. цепочка над алфавитом Σ_{del} с длиной, равной 0 или 1.

Понятие *аббревиатуры* определим следующим образом: строка *a* является аббревиатурой или сокращением словосочетания *st*, тогда и только тогда, когда строку *a* можно представить в виде конкатенации сокращений слов словосочетания *s*, т.е.

$$a = p_1 \bullet p_2 \bullet \ldots \bullet p_k, \quad \forall p_i F(p_i, w_i). \quad (3)$$

Для создания критерия похожести строк следует учитывать, что как полное наименование, так и аббревиатуры и сокращения, сознательно используемые оператором, могут быть подвержены искажениям, вызванным ошибкой человека. Поэтому представляется целесообразным ввести понятие *расстояния редактирования аббревиатур* между двумя строками, определяемого минимальным количеством операций вставки или удаления символа, необходимых для такого преобразования одной из строк, после которого она будет являться аббревиатурой или сокращением второй строки. Тогда *критерием похожести двух строк* должно стать условие, при котором определенное выше расстояние между двумя строками не превышает некоторый заданный порог.

Постановка задач

Теперь, базируясь на таком критерии похожести, приступим к созданию метода поиска похожих строк. Здесь, однако, следует отметить, что в зависимости от этапа, на котором будет производиться поиск, имеют место некоторые отличия в постановке задачи. Так, при вводе данных, когда поиск применяется «на лету» с целью избежать появления дублирующихся строк, постановка задачи будет выглядеть следующим образом.

Пусть дан набор строк $ST=\{st_1, st_2, ..., st_n\}$ и строка поиска *src*. Требуется найти все строки $st \in ST$, для которых выполняется условие:

$$d_a(st, src) \le \lambda, \quad (4)$$

где $d_a(st, src)$ – расстояние редактирования аббревиатур между строками *st* и *src*; λ – некоторый порог похожести.

В случае же интеграции данных из неоднородных источников нам требуется найти все пары строк $st_1, st_2 \in ST$, для которых выполняется условие:

$$d_a(st_1, st_2) \le \lambda. \quad (5)$$

Зачастую для первого случая набор строк представляет собой реквизиты медленно изменяющегося справочника, и предварительная обработка списка является допустимой.

В то же время в первом варианте практика накладывает значительно более жесткие требования к времени поиска. Так, для того, чтобы не прерывать работу оператора, время поиска не должно превышать 1 – 3 секунд. В данной работе рассматриваются оба случая следующим образом. В следующих разделах предлагается метод поиска пар похожих строк в соответствии с (5). Далее обсуждаются изменения, вносимые для создания эффективного метода в соответствии с (4), с учетом отличий, представленных выше.

Расчет расстояния редактирования аббревиатур

Для расчета расстояния редактирования аббревиатур в данной работе используется следующий подход. На первом этапе для полного наименования (или более длинного) st_1 длиной r символов строим недетерминированный конечный автомат (НКА), который допускает цепочки, являющиеся аббревиатурой или сокращением наименования в соответствии с (3):

$$A_e = (Q, \Sigma, \delta, q_0, F),$$

где $Q = \{q_0, q_1, q_2, ..., q_r\}$ – множество состояний такого автомата, причем, если автомат находится в q_i, $i = \overline{1,r}$, то это можно интерпретировать следующим образом: “Цепочка, поступившая на вход автомата, является аббревиатурой или сокращением строки $st_1' = s_1 \bullet s_2 \bullet ... \bullet s_i$ и, следовательно, строки st_1”; Σ – конечное множество входных символов (для алфавита $\Sigma_c \cup \Sigma_{del}$); q_0 – начальное состояние; $F = \{q_1, q_2, ..., q_r\}$ – множество допускающих состояний; δ – функция переходов автомата, полученная в соответствии со следующими правилами:

а) необходимо обеспечить переходы по которым, если на автомат подается цепочка равная полному наименованию, попадаем в состояние q_1, т.е. $q_i \in \delta(q_{i-1}, s_r)$;

б) необходимо обеспечить переходы, предусматривающие варианты сокращения слов в соответствии с (2).

Пример такого НКА, построенного для словосочетания “Привет мир”, представлен на рис. 1.

На втором этапе путем модификации построенного ранее НКА, получим автомат, допускающий все цепочки, для которых расстояние редактирования аббревиатур не превышает некоторого λ:

$$A = (Q, \Sigma, \delta, q_0, F),$$

где $Q = \{q_{0,0}, q_{0,1}, q_{0,2}, ..., q_{0,r}, ..., q_{\lambda,0}, q_{\lambda,1}, q_{\lambda,2}, ..., q_{\lambda,r}\}$ – множество состояний такого автомата, причем, если автомат находится в $q_{j,i}$, $j = \overline{0,\lambda}, i = \overline{1,r}$, то это можно интерпретировать следующим образом: “Существует путь редактирования длиной в j операций, после выполнения которых, цепочка, поступившая на вход автомата, будет являться аббревиатурой или сокращением строки $st_1' = s_1 \bullet s_2 \bullet ... \bullet s_i$ и, следовательно, строки st_1”. Соответственно, если НКА находится во множестве состояний

$$States = \{q_{f1,s1}, q_{f2,s2}, ..., q_{fh,sh}\},$$

то расстояние редактирования аббревиатур между исходной строкой s_1 и строкой, поданной на вход автомата, будет равным $d_a = \min(f1, f2, ..., fh)$; Σ – конечное множество входных символов (алфавит $\Sigma_c \cup \Sigma_{del}$); $q_{0,0}$ – начальное состояние; $F = \{q_{0,1}, q_{0,2}, ..., q_{0,r}, ..., q_{\lambda,1}, q_{\lambda,2}, ..., q_{\lambda,r}\}$ – множество допускающих состояний; δ – функция переходов автомата, полученная в соответствии со следующими правилами:

а) необходимо обеспечить переходы, аналогичные переходам автомата A_e между состояниями, находящимися на одном уровне, т.е.

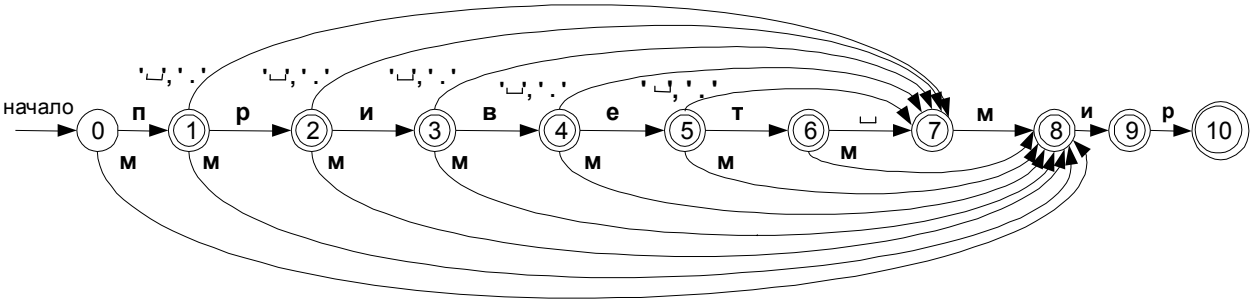

Рис. 1. Диаграмма переходов НКА, допускающего цепочки, являющиеся сокращением или аббревиатурой словосочетания «привет мир»

замены символа трактуется как две операции) до осей:

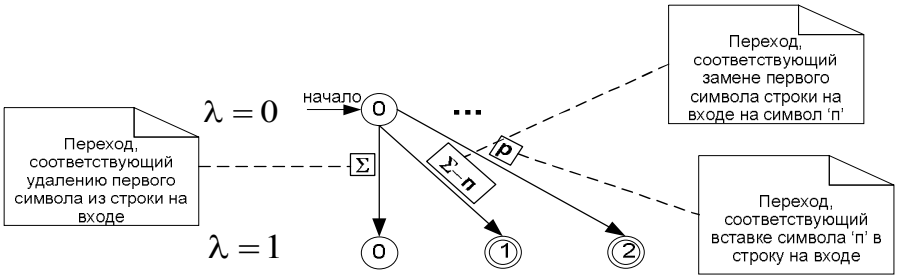

Рис. 2. Переходы, отражающие операции редактирования

$$\exists\delta_{A_e}(q_i,s)=\{q_{x1},...,q_{xm}\}, i=\overline{0,r}, s\in\Sigma \Rightarrow$$
$$\Rightarrow \delta_A(q_{j,i},s)\supseteq\{q_{j,x1},...,q_{j,xm}\}, \forall j=\overline{0,\lambda};$$

б) необходимо обеспечить переходы, отражающие операции вставки, удаления и замены символов между состояниями, находящимися на разных уровнях (рис. 2).

Теперь, имея в распоряжении метод расчета расстояния редактирования аббревиатур, мы можем предложить «наивный» метод решения задачи поиска похожих пар строк в списке. В соответствии с «наивным» методом, мы для каждой строки списка должны проверить, выполняется ли условие (4). Очевидно, что в связи с тем, что будет выполнено $n\cdot(n–1)$ таких проверок, при достаточно большом n алгоритм решения задачи будет чрезвычайно неэффективным по быстродействию.

Поэтому необходимо создание такого метода, который бы значительно превышал по показателям быстродействия «наивный» метод, достигая при этом таких же результатов.

Быстрый метод поиска пар похожих строк

Подобно решению, предложенному в [6], в данной работе решение состоит из двух шагов. На первом шаге все элементы набора *ST* отображаются в k-мерное Евклидово пространство E^k, с осями которого ассоциируются $o_1,o_2,...,o_k$, $(k<n)$ – k выбранных случайным образом элементов набора *ST*, т.е. каждому элементу $st_i\in ST$ ставится в соответствие точка k-мерного Евклидова пространства $P(st_i)$, координаты которой равны простым расстояниям Левенштейна (операция

$$P(st_i)_j=d_s(st_i,o_j), i=\overline{1,n}, j=\overline{1,k}.$$

Далее на следующем шаге рассматриваются лишь те пары строк которые отвечают строго доказанным необходимым условиям похожести. Среди отличий предлагаемого метода от метода в упомянутой работе следует отметить, что в связи с отличием используемых критериев похожести строк полностью отличны необходимые условия похожести. Поэтому, прежде чем приступить к изложению сути предлагаемого метода сформулируем ряд строго доказанных математических утверждений.

Утверждение 1. Если строка st_1 является аббревиатурой или сокращением строки st_2, то наибольшая общая подпоследовательность (longest common subsequence) между строками st_1 и st_2 будет равна st_1.

Утверждение 2. Если одну из строк st_1 или st_2 исказить одной операцией редактирования, то величина *MyLCS* не увеличится более чем на 1, т.е.

$$MyLCS(st_1^*,st_2^*)\le MyLCS(st_1,st_2)+1,$$

где $MyLCS(x,y)=\min(|x|,|y|)-|lcs(x,y)|$, st_1^*,st_2^* – значения исходных строк после произведенной операции редактирования.

Утверждение 3. Расстояние редактирования аббревиатур между строками st_1 и st_2 не превышает величину некоторого порога λ тогда и только тогда, когда величина $MyLCS(st_1,st_2)$ также не превышает заданного порога, т.е. $d_a \ge MyLCS(st_1,st_2)$.

Утверждение 4. Если st_1, st_2 – строки, расстояние редактирования аббревиатур между

которыми не больше некоторого порога λ, то точка $P(st_2)$ размещается в E^k в пределах гиперкуба с центром в $P(st_1)$ и стороной $2(2\lambda + \|st_1| - |st_2\|)$.

Утверждение 5. Если st_i, st_j – строки, расстояние редактирования аббревиатур между которыми не больше некоторого порога λ, то абсолютное значение разности расстояний от точек $P(st_i)$ и $P(st_j)$ до начала координат в E^k не превышает

$$(2\lambda + \|st_1| - |st_2\|)\sqrt{k},$$

т.е. $|\rho(P(st_i),0) - \rho(P(st_j),0)| \le (2\lambda + \|st_1| - |st_2\|)\sqrt{k}$.

Теперь опишем подробнее суть выполняемых действий на каждом из этапов предлагаемого решения.

1. Как уже было сказано выше, на первом этапе происходит отображения исходного набора ST в k-мерное Евклидово пространство E^k. Здесь следует лишь отметить, что структура данных, используемая для хранения точек пространства, должна обеспечивать быстрый последовательный доступ ко всем точкам, для которых заданы значения расстояния до начала координат и длины исходной строки из набора *ST*.

2. На втором этапе для каждой точки $P(st_i)$, варьируя значения переменной *len* от минимального значения длины строки до максимального, в соответствии с утверждением 5 просматриваем точки, для которых расстояние до начала координат находится в диапазоне $[\rho(P(st_i),0) - (2\lambda + \|st_i| - len|)\sqrt{k};$. $\rho(P(st_i),0) + (2\lambda + \|st_i| - len|)\sqrt{k}]$. Каждая рассматриваемая точка, в свою очередь, проверяется в соответствии с утверждением 4 на попадание в пределы гиперкуба с центром в точке $P(st_i)$. Далее для строк, соответствующих точкам, оказавшихся в пределах гиперкуба, рассчитывается величина *MyLCS*. И, наконец, в соответствии с утверждением 3, только в случае, если *MyLCS* не превышает порогового значения, производится "дорогое" вычисление расстояния редактирования аббревиатур.

Быстрый метод поиска строк, похожих на заданную

Предложенный ранее метод сравнительно легко преобразовать для поиска в наборе строк, похожих на заданную. Отличие в этом случае будет состоять лишь в том, что на втором этапе поиск будет только для заданной строки *src*. В то же время, возможность предварительной обработки набора строк позволяет предусмотреть следующие меры, направленные на повышение скорости поиска.

1. Отображение списка в k-мерное Евклидово пространство E^k может быть получено заранее, и лишь модифицироваться при добавлении или удалении строк из *ST*. Таким образом, исключается время, необходимое для выполнения $k \cdot n$ расчетов расстояний Левенштейна.

2. Несмотря на то, что количество «дорогих» вычислений расстояний редактирования аббревиатур существенно сокращается относительно «наивного» подхода, они все равно оказывают значительное влияние на время выполнения поиска в целом. В то же время, воспользовавшись тем, что из любого НКА можно получить эквивалентный ему детерминированный конечный автомат (ДКА) [8], можно существенно сократить время выполнения одной проверки. И если в методе поиска пар похожих строк такое преобразование было бы неоправданным в связи с невысокой скоростью алгоритма получения ДКА, то в данном случае ДКА для всех строк из набора *ST* строятся заранее, что позволяет повысить быстродействие метода поиска строк, похожих на заданную.

Экспериментальные исследования полученных методов

Экспериментальные исследования описанных методов были проведены на основе списка о занимаемых сотрудниками должностях, состоящего из 895 строк, полученного в отделе кадров университета «ХАИ». Все эксперименты проводились на персональном компьютере с

процессором CELERON 566 МГц и 196 Мб ОЗУ. Операционная система – Windows NT 4 Server, компилятор – Borland Delphi 6. При выполнении наивного и предлагаемого метода поиска пар похожих строк в обоих случаях было найдено 7163 пары похожих наименований. Выявлены такие похожие строки, как «головний бухгалтер», «гол. бухгалтер» (d_a=0); «старший преподаватель», «ст. преподаватель» (d_a=1);. «инженер-программист», «програміст» (d_a=2). В то же время, при оценке времени выполнения методов было выявлено превосходство предлагаемого подхода перед «наивным». Рис. 3 иллюстрирует зависимость времени выполнения методов от количества строк в списке.

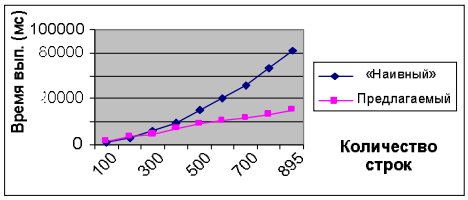

Рис. 3. Результаты экспериментальных исследований методов поиска пар похожих строк

Оценка предлагаемого метода поиска строк, похожих на заданную, производилась следующим образом. В качестве заданной строки выбиралась одна из строк упомянутого списка. Для выбранной строки выполнялись методы поиска похожих строк. Для оценки быстродействия методов использовалось среднее время поиска для всех возможных строк из списка. Так, в случае, когда использовался метод, не предусматривающий предварительную обработку списка, среднее время выполнения одного поиска составило 632 миллисекунды. Предварительная обработка в соответствии с п. 5 позволила снизить этот показатель до 145 миллисекунд.

Заключение

Таким образом, получен новый критерий, позволяющий находить строки, представляющие одну и ту же сущность реального мира, и искаженные относительно друг друга ошибками оператора и использованием сокращений и аббревиатур. Представленные необходимые условия похожести строк позволяют ограничить область поиска и, следовательно, получить эффективные методы. Эффективность предлагаемых методов подтверждается проведенными экспериментами.

Литература

1. Maletic, J.; Marcus, A. Data Cleansing: Beyond Integrity Analysis // Proceedings of The Conference on Information Quality (IQ2000). – MIT, Boston, MA, USA. – 2000. – P. 200-209.

2. Информационно-аналитические модели управления техническими высшими учебными заведениями / А.Н. Гуржий, В.С. Кривцов, А.С. Кулик и др. – Х.: Нац. аэрокосм. ун-т «ХАИ», 2004. – 387 с.

3. Кнут Д. Искусство программирования для ЭВМ: В 3 т. – М.: Мир, 1978. – Т 3. Сортировка и поиск. – 844 с.

4. Approximate string joins in a database (almost) for free / L. Gravano, G. Panagiotis // Proceedings of the VLDB Conference. – 2001. – P. 491-500.

5. Baeza-Yates R., Navarro G. A practical index for text retrieval allowing errors // Proc. of the XXIII Latin American Conf. on Inf. (CLEI'97). – 1997. – P. 273-282.

6. Кулик А.С., Чухрай А.Г. Метод обнаружения «похожих» наименований номенклатуры в неоднородных справочниках технико-экономической информации вуза // Открытые информационные и компьютерные интегрированные технологии: – Х.: Нац. аэрокосм. ун-т «ХАИ», 2003. – Вып. 17. – С. 147-152.

7. A. Monge Matching Algorithms within a Duplicate Detection System // IEEE Techn. Bulletin Data Engineering. – 2000. – V. 23, №. 4. – P. 14-20.

8. Хопкрофт Д.Э., Мотвани Р., Ульман Дж.Д. Введение в теорию автоматов, языков и вычислений. – М.: Изд. дом "Вильямс", 2002. – 528 с.

Поступила в редакцию 23.02.2006

Рецензент: д-р техн. наук, проф. В.М. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков.